

# Hybrid Mobile Applications - ICD0018

TalTech IT College, Andres Käver, 2021-2022, Fall semester

Email: [andres.kaver@taltech.ee](mailto:andres.kaver@taltech.ee)



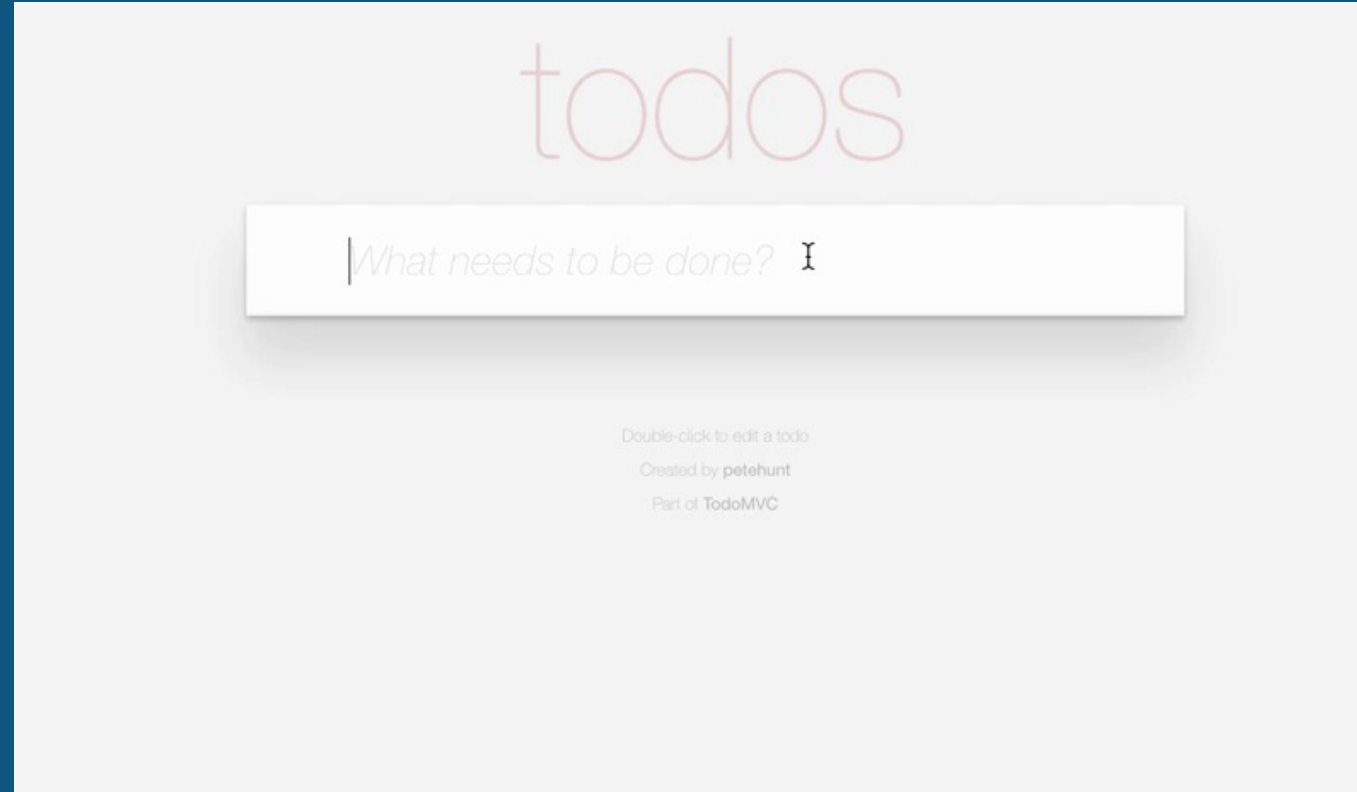
- ▶ Still to come... maybe....
- ▶ Persistence
  - ▶ Networking
    - ▶ **Tokens/Auth**, Json parsing in background, WebSockets/SignalR
  - ▶ **Working with SQLite**
  - ▶ Files, Key-Value data
- ▶ Cupertino vs Material
- ▶ Background services (Isolate), Media, notifications, push, sensors
- ▶ Plugins – interfacing with native API's
- ▶ i18n, accessibility
- ▶ Animations, Assets

# Flutter – final project

3

## ▶ ToDo App

- ▶ State and SQLite
- ▶ Sync to ASP.NET Core 3 backend (provided by course)
  - ▶ Auth
- ▶ Lists, dragging, swiping



- ▶ <https://medium.muz.li/designing-pocket-lists-18b6cafd1161>
- ▶ Backend source code: <https://git.akaver.com/taltech-public/com.akaver.taltech>
- ▶ <https://taltech.akaver.com> <https://taltech.akaver.com/swagger/>

# Flutter – final project

4

- ▶ Features to implement
  - ▶ Create/edit todos (category, priority)
  - ▶ Delete todo by swiping
  - ▶ Mark done/not done in list
  - ▶ Filter todo list by status (all/done/not done)
  - ▶ Change tasks order via drag-drop
  - ▶ Change theme (light to dark and vice versa) at runtime
  - ▶ Login/logout/sync data
  - ▶ Register new account
  - ▶ Work offline (local db)

# Flutter - Auth

5

- ▶ Flow – Register, Login, Work, Logout
  - ▶ Content-Type: application/json
  - ▶ Host: <https://taltech.akaver.com/>

# Flutter – Auth Register

6

- ▶ Register

- ▶ /api/v1/account/register

- ▶ Method: post

- ▶ Body

- {  
 "email": "test01@akaver.com",  
 "password": "Kala.maja1"  
}

- ▶ Response

- {  
 "token": "eyJhbGciOiJIUzI...",  
 "status": "Account created for test01@akaver.com"  
}

# Flutter – Auth Login

7

- ▶ Login

- ▶ /api/v1/account/login

- ▶ Method: post

- ▶ Body

- {  
 "email": "test01@akaver.com",  
 "password": "Kala.maja1"  
}

- ▶ Response

- {  
 "token": " eyJhbGciOiJIUzI...",  
 "status": "Logged in"  
}

# Flutter – Auth Logout

8

- ▶ Logout
  - ▶ Just delete the token



# Flutter – api endpoints

9

- ▶ ToDo Api endpoints (get all)
  - ▶ /api/v1/todocategories
  - ▶ /api/v1/todopriorities
  - ▶ /api/v1/todotasks
- ▶ /api/v1/todocategories/1
- ▶ Get, Post, Put, Delete – classical Rest backend http verbs

# Flutter - Requests with auth

10

- ▶ On every Rest API request

- ▶ Header:

Authorization: bearer eyJhbGciOiJIUzI...

# Flutter – Rest Auth

11

## ► Login/Register

```
Future<JWT> fetchToken(String email, String password) async {  
    Map<String, String> headers = {"Content-type": "application/json"};  
  
    final response = await http.post(  
        'https://taltech.akaver.com/api/v1/account/login',  
        headers: headers,  
        body: '{"email": "$email", "password": "$password"}');  
  
    if (response.statusCode == 200) {  
        return JWT.fromJson(json.decode(response.body));  
    } else {  
        return JWT(token: "", status: response.reasonPhrase);  
    }  
}
```

## ▶ JWT class

```
import 'package:flutter/widgets.dart';

class JWT {
  final String token;
  final String status;

  JWT({@required this.token, @required this.status});

  factory JWT.fromJson(Map<String, dynamic> json){
    return JWT(token: json['token'], status: json['status']);
  }
}
```

# Flutter – App State

13

- ▶ Wrap the whole app inside state provider (including routes)

```
void main() {  
  runApp(  
    ChangeNotifierProvider<AuthModel>(  
      builder: (_) => AuthModel(),  
      child: MaterialApp(  
        title: 'Navigation',  
        initialRoute: '/',  
        routes: {  
          '/': (context) => LoginScreen(),  
          '/main': (context) => MainScreen(),  
        },  
      ),  
    ),  
  );  
}
```

# Flutter – DataBase SQLite

14

- ▶ Packages
  - sqlite: ^1.3.1+1
  - path: ^1.7.0
  - path\_provider: ^1.6.21
- ▶ Implement database helper class

- ▶ Singleton
- ▶ Lazy

```
class DatabaseHelper {  
    static final _databaseName = "MyDatabase.db";  
    static final _databaseVersion = 1;  
  
    DatabaseHelper._internal();  
    static final DatabaseHelper instance = DatabaseHelper._internal();  
    factory DatabaseHelper(){  
        return instance;  
    }  
  
    static Database _database;  
    Future<Database> get database async {  
        if (_database != null) return _database;  
        _database = await _initDatabase();  
        return _database;  
    }  
  
    _initDatabase() async {  
        Directory documentsDirectory = await getApplicationDocumentsDirectory();  
        String path = join(documentsDirectory.path, _databaseName);  
  
        return await openDatabase(path,  
            version: _databaseVersion, onCreate: _onCreate);  
    }  
}
```

## ► \_onCreate

```
static final table = 'my_table';

static final columnId = '_id';
static final columnName = 'name';
static final columnAge = 'age';

// SQL code to create the database table
Future _onCreate(Database db, int version) async {
  await db.execute('''
    CREATE TABLE $table (
      $columnId INTEGER PRIMARY KEY,
      $columnName TEXT NOT NULL,
      $columnAge INTEGER NOT NULL
    )
  ''');
}
```



# Flutter – SQLite, data access

17

```
// Inserts a row in the database where each key in the Map is a column name
// and the value is the column value. The return value is the id of the
// inserted row.
```

```
Future<int> insert(Map<String, dynamic> row) async {
    Database db = await instance.database;
    return await db.insert(table, row);
}
```

```
// All of the rows are returned as a list of maps, where each map is
// a key-value list of columns.
```

```
Future<List<Map<String, dynamic>>> queryAllRows() async {
    Database db = await instance.database;
    return await db.query(table);
}
```

```
// All of the methods (insert, query, update, delete) can also be done using
// raw SQL commands. This method uses a raw query to give the row count.
```

```
Future<int> queryRowCount() async {
    Database db = await instance.database;
    return Sqflite.firstIntValue(
        await db.rawQuery('SELECT COUNT(*) FROM $table'));
}
```

# Flutter – SQLite, data access

18

- Implement repo probably!

```
// Assume that the id column in the map is set. The other
// column values will be used to update the row.
Future<int> update(Map<String, dynamic> row) async {
  Database db = await instance.database;
  int id = row[columnId];
  return await db.update(table, row, where: '$columnId = ?', whereArgs: [id]);
}

// Deletes the row specified by the id. The number of affected rows is
// returned. This should be 1 as long as the row exists.
Future<int> delete(int id) async {
  Database db = await instance.database;
  return await db.delete(table, where: '$columnId = ?', whereArgs: [id]);
}
```

# Flutter – More?

19

- ▶ Still to come... maybe....
- ▶ Persistence
  - ▶ Networking
    - ▶ Json parsing in background, WebSockets/SignalR
  - ▶ Files, Key-Value data
- ▶ Cupertino vs Material
- ▶ Background services (Isolate), Media, notifications, push, sensors
- ▶ Plugins – interfacing with native API's
- ▶ i18n, accessibility
- ▶ Animations, Assets